

# The MetaboX library: building metabolic networks from KEGG database

F. Maiorano<sup>1\*</sup>, L. Ambrosino<sup>2</sup>, M.R. Guarracino<sup>1</sup>

## Abstract

**Motivation** In many key applications in the field of metabolomics, such as toxicology or nutrigenomics, it is of interest to profile and detect physiological changes in metabolic pathways. For this purpose, it is useful to build a comprehensive graphical representation which shows metabolic processes occurring in an organism using networks. To model these systems it is possible to describe both reactions and relations among enzymes and metabolites. In this way, analysis of possible changes or perturbations impact throughout the network are easier to understand, detect and predict. To address this problem, we develop a library to build metabolic networks starting from a list of compounds.

**Results** We release the MetaboX library, an open source PHP framework for developing metabolic networks from a set of compounds. This library provides easy access to the *Kyoto Encyclopedia for Genes and Genomes* (KEGG) database using its RESTful *Application Programming Interfaces* (APIs), and methods to enhance manipulation of the information returned from KEGG webservice. MetaboX includes methods to extract information about a resource of interest (e.g. metabolite, reaction and enzyme) for further processing and storing purposes. MetaboX is modular, thus developers can contribute with alternative implementations or extensions. Each component of the library is designed with minimum dependency on other components. Supplementary information available in Table 1.

**Availability** The MetaboX library is available under the AGPL license on gitHub repository (<https://github.com/Gregmayo/MetaboX-Library>)

## Keywords

Metabolomics — Network construction — Data mining

<sup>1</sup> *Laboratory for Genomics, Transcriptomics and Proteomics, Institute for High-Performance Computing and Networking, National Research Council, Via P. Castellino 111, 80131 (NA), Italy*

<sup>2</sup> *Dept. of Agricultural Sciences, University of Naples Federico II, Via Università 100, 80055 Portici (NA), Italy*

\*Corresponding author: francesco.maiorano@na.icar.cnr.it

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Implementation</b>	<b>3</b>
2.1	Network Construction . . . . .	3
2.2	Reactants Network . . . . .	4
2.3	Bipartite Enzyme-Metabolite Network . . . . .	4
2.4	Unipartite Enzymes Network . . . . .	5
2.5	Data Export . . . . .	5
<b>3</b>	<b>Results and Discussion</b>	<b>5</b>
<b>4</b>	<b>Conclusions</b>	<b>7</b>
	<b>References</b>	<b>8</b>

## 1. Introduction

In metabolomics applications it is often of interest to model relationships and interactions among compounds and enzymes, such as protein-protein interactions, metabolite pathways or pathway flows. One of the main challenges in metabolomics is to model these interactions in the form of networks [HU *et al.* 2011, Cloots *et al.* 2011]. Such networks make it

easier to understand the topological and functional structure of molecules and their interactions. Furthermore, once the network has been built, various statistics can be obtained for characterization and comparison. Indeed, a network represents a convenient way to model objects and their relationships as complex systems. Modeling a network of metabolites provides several ways to further analyze types of interactions, to understand the role of each metabolite in a particular pathway and to detect changes. The problem we address is to build metabolite *reaction*, *unipartite enzymes* and *bipartite enzyme-compound* networks, starting from a list of compounds and information on metabolism gathered from a database. These networks models were used in other research studies in order to identify so-called reporter metabolites [Raosaheb *et al.* 2005]. In a metabolite reaction network, two metabolites are connected if they are known to react in the same reaction. In a unipartite enzyme network, two enzymes are connected if they share at least one metabolite in the reactions they catalyze. In a bipartite enzyme-compound network, each enzyme is connected to every metabolite that is present in the reactions it catalyzes. There are several publicly available databases that

store and distribute metabolic information on molecular compounds. Among these we cite Metacyc [Krieger *et al.* 2004], EcoCyc [Keseler *et al.* 2013], HMDB [Wishart *et al.* 2007], Lipid Maps [Sud *et al.* 2007], BioCyc [Karp *et al.* 2005], Reactome [Matthews *et al.* 2009], PubChem [Wang *et al.* 2012], Chebi [Hastings *et al.* 2013], ChemSpider [Pence *et al.* 2010], IIMDB [Menikarachchi *et al.* 2013], Meltin [Smith *et al.* 2005], and KEGG [Kanehisa *et al.* 2011]. It is out of the scope of this paper to describe the characteristics of all these databases, and we focus our attention on the latter, because it is the most up-to-date and comprehensive one. KEGG stands as a *de facto* standard, a database of high-level functions and utilities for several biological organisms. It is possible to query the database with a web interface using one compound, and obtain information on the reactions in which it is involved, the stoichiometric equations, enzymes that catalyze the reaction, and metabolic pathways in which these reactions are involved. Its website graphically displays the stored pathways, but there is no functionality to aggregate multiple reactions starting from multiple compounds, neither it is possible to search using a set of compounds. On the other hand, it is possible to programmatically query the database, obtaining such information in form of flat files. For large lists of input nodes it can be very difficult to manually gather information of interest to build a network, as well as other metadata useful to get a complete understanding of the biological system. To overcome these difficulties, some software exist, and partially solve these problems. MetaboAnalyst [Xia *et al.* 2012] provides a web-based analytical pipeline for metabolomic studies. Its web interface can be used to load data as a list, for statistical analysis, as well as pathway analysis. When queried with a list of compounds, it returns information on pathways taken from KEGG but, for reasons related to XML representation of KEGG pathways, information concerning reactions and substrates are partially lost. Therefore, the resulting metabolic network is often disconnected and it does not represent a good model for graph analysis. The source code is not available, neither it provides APIs of any form. INMEX [Xia *et al.* 2013], introduces an integrative meta-analysis of expression data and a web-based tool to support meta-analysis. It provides a web interface to perform complex operations step-by-step. While it supports custom data processing, annotation and visualization, it does not provide any APIs to extend core functionalities and it cannot be deployed in a custom environment. Finally, MetaboLyzer [Mak *et al.* 2013] implements a workflow for statistical analysis of metabolomics data. It aims at both simplifying analysis for investigators who are new to metabolomics, and providing the flexibility to conduct sophisticated analysis to experienced investigators. It uses KEGG, HMDB, Lipid Maps, and BioCyc for putative ion identification. However, it is specifically suited for analysis of post-processed liquid chromatography-mass spectrometry (LC-MS)-based metabolomic data sets. Although these software give the possibility to model a new network starting from a list of compounds, providing relevant tools for statistical and

functional analyses, they miss the capability to programmatically query metabolomics resources in order to develop novel applications. With the aim to fill that gap, we introduce the MetaboX library. With respect to the tools presented above, MetaboX provides a framework to model custom network layouts from a list of input nodes. Based on the nature of input nodes, we provide a set of classes to gather related information and programmatically build a network. MetaboX is an open source library that aims to get a growing community of researchers and developers to support metabolomic analysis. With the MetaboX library, developers are able to model a network in different ways using the available methods to create a custom network layout that meets their needs. The library has been built with the possibility to extend data information gathering, such as downloading these from databases other than KEGG or merge information collected from multiple databases. The library design is modular, with the aim to give developers an exhaustive framework to implement different types of network builders from lists of compounds. Thus, information-gathering and interactions-detection tasks are completed programmatically, with speed and consistency benefits and the possibility to work with large lists of nodes. In the network construction process, MetaboX handles the following steps:

- *connect* to the resource provider database using the PHP `libcurl` library (<http://curl.haxx.se/libcurl/>). It connects and communicates with servers using different types of protocols.
- *query* a resource provider using methods to retrieve nodes and interactions. As KEGG does not provide a structured query response, we built a translation layer to extract information from flat files.
- *extract* requested resource attributes from returned data, parsing and storing them. This task is achieved using regular expressions.
- *cache* resource attributes to file using a convenient data structure for serialization and for sharing and processing purposes such as JSON (<http://www.json.org>).
- *build* a consistent data structure with information about requested resources using all previous steps, in order to build a network from collected data. Results consist of a weighted edgelist and a list of network nodes. It also includes specific resource information.

In the *connect* step, MetaboX currently supports HTTP, HTTPS, FTP, and ldap protocols. It also supports HTTPS certificates, POST, PUT and FTP uploading, which are natively available in the `curl` library (<http://php.net/manual/en/book.curl.php>). It is also possible to *query* the KEGG database with a list of resources of interest and then parse the response to firstly separate information about each one and then extract specific data. In the *extract* step downloaded data are parsed to produce new files ready for next steps. We locally *cache* data to

load resource information every time it is requested again by a new process. We design the caching system for MetaboX to speed up computation and to produce a sustainable amount of requests to the resource provider system. It is possible to invalidate the cache in order to reload updated data, and to manually delete the cache so that the library can update it. Finally, the *build* step is intended to put together all gathered information and output the resulting network. Every build method connects two nodes differently in each network model, that is *Reactant Graph* implementation of the build method is different from both *Enzyme Unipartite Graph* and *Enzyme Bipartite Graph*. In the following section, we report the implementation of the MetaboX library, detailing how it provides easy information access and data manipulation. We explain how to use the library to build the different networks and how to export the result for further analysis with tools like Cytoscape [Cline *et al.* 2007] (<http://www.cytoscape.org>). Then, in Section 3, we provide a case study and discuss results. Finally, in Section 4, we conclude, providing details on future work directions and open problems.

## 2. Implementation

KEGG used to expose SOAP APIs to standard software, such as Matlab Bioinformatics toolbox, in order to provide pathway representation and visualization. Since SOAP APIs were suppressed on 31st december 2012, these tools may not work anymore (<https://www.biocatalogue.org/announcements/37>).

```

1 // Retrieve and collect compound information
2 foreach( $compounds as $compound ){
3     $_cpd_id = trim($compound);
4     $cpd_loader = new MetaboX\Resource\Loader\
5         Compound($_cpd_id, $cpdLoaderConfig);
6     $_compounds[$_cpd_id] = $cpd_loader->load();
7 }
8 // Retrieve and collect reactions information
9 foreach($_compounds as $id => $compound){
10     $rn_list = $compound->reactionIdCollection;
11
12     if( $rn_list ){
13         foreach( $rn_list as $rn ){
14             $_rn_id = trim($rn);
15             $rn_loader = new MetaboX\Resource\Loader\
16                 Reaction($_rn_id, $rnLoaderConfig);
17             $_reactions[$_rn_id] = $rn_loader->load();
18         }
19     }
20 }
21 // Create reactants graph
22 $_graph = new MetaboX\Graph\ReactantsGraph(
23     $_reactions);
24 $_graph->build($compounds);

```

**Listing 1.** Loading Metabolites and Reactions metadata from KEGG

At the moment of writing, KEGG only offers a RESTful API interface thus MetaboX is designed to query these in an appropriate manner. KEGG returns plain text upon web-service calls, thus making it necessary to parse results and

arrange them in a data structure. We query KEGG multiple times and store the gathered information to file. The file format we use is JSON which is a lightweight data-interchange format, human-readable and writable. JSON is a text format that is completely language independent but uses conventions that are familiar to C-family programmers. These properties make JSON an ideal data-interchange language. To limit requests to KEGG, the MetaboX library loads previously processed resources from local storage, if they are available. A sample request for a resource in KEGG can be achieved using the following url: <http://rest.kegg.jp/<operation>/<argument>>. For instance, to retrieve information about metabolite C01290, we use <http://rest.kegg.jp/get/cpd:C01290>. We provide details about the implementation in the following sections.

### 2.1 Network Construction

We deal with compounds, reactions and pathways. To handle such a variety of entities the MetaboX library contains classes to instantiate them. *AbstractResourceLoader* is an abstract class that provides methods needed to load an entity. To model an entity with a new class, this has to extend the abstract class and implement the abstract *load* method. When an entity is instantiated, this method first checks for existing records in the cache. If the requested entity has not been processed previously, a new file is built upon KEGG response. This pattern is used to load metabolites, reactions, pathways and enzymes. We provide several helper methods to extract information about resources from plain text using regular expressions. When the entity has been successfully processed, we serialize it to file for further reference. The attributes that define a metabolite are: *id*, *formula*, *exact mass*, *molecular weight*, *reaction list*, *pathway list*, *enzyme list*. Lists of other entities of interest that are related to a metabolite, such as reactions, pathways and enzymes, are loaded with different API call. For instance, if the *load* of C01290 returns a list of 10 reactions (<http://rest.kegg.jp/get/cpd:C01290>), we use a RESTful url (<http://rest.kegg.jp/get/rn:RNXXXX>) to instantiate each of these reactions. For reactions, we collect *id*, *name*, *definition*, *equation*, *enzymes* and *pathways*. For data manipulation purposes and to conveniently organize reaction information about input metabolites, we process reaction equations separating reactants from products in a data structure. Cache directories can be set in a configuration file. Each resource is stored in a dedicated resource directory and files are named after resource id (e.g. {resource}/{resource.id}.json - compound/C00002.json). The configuration file 'config.ini' is divided in sections and it is possible to specify storage directories for entities (e.g. config->directory->compound or config->directory->reaction) as well as KEGG API urls (e.g. config->url->compound or config->url->reaction). This approach is helpful if the entities become available in different urls or from another resource provider. In the MetaboX library we provide an interface to build several networks. *AbstractGraphBuilder* is an abstract class that defines the general structure of the resulting network. Specific network builder classes

must implement the abstract *build* method provided in the abstract builder which takes one optional parameter. This is a list of metabolites out of which a sub network has to be built. To create a new type of network, a builder class should provide the construction of a network involving input metabolites and others involved in common reactions, or other entities, such as enzymes. If the optional parameter is specified, the builder method should create a network with set of nodes given by input parameter. When the network-construction process is completed, *getGlobalGraph* and *getSubGraph* methods return a multidimensional array containing the list of nodes, a weighted edgelist, where the weight represents the number of times a reaction has been found, and the list of connected and not connected nodes, in the case of a sub network.

## 2.2 Reactants Network

A network of reactants  $G = (V, E)$  is an undirected graph where each node represents a metabolite and two given nodes  $A$  and  $B$  in  $V$  interact with each other only if there is at least one reaction equation where  $A$  and  $B$  are involved as reactants. *ReactantsGraph* class builds a network out of a list of metabolites. To achieve this task, we first gather metabolites and reactions data from KEGG (Listing 1). We create a list of reactions that involve input metabolites and pass it to the class. In this case, the *build* method cycles through the list of reactions and, for each one, the list of substrates is extracted. We then connect each substrate to one another and when all direct network interactions have been built, we produce a weighted edgelist. Such edgelist represents a network including input compounds and all other compounds involved in processed reactions. We also save a weighted list of interactions that only include input compounds, this resulting in a smaller network which can be seen as a sub network of the global weighted interaction list. As shown in Fig. 2, the sub network is embedded in the global one. A builder class exposes methods to compute results and pass them to the graph writer classes in order to produce a file format that is suitable to the needs of further analysis, such as SIF and XML.

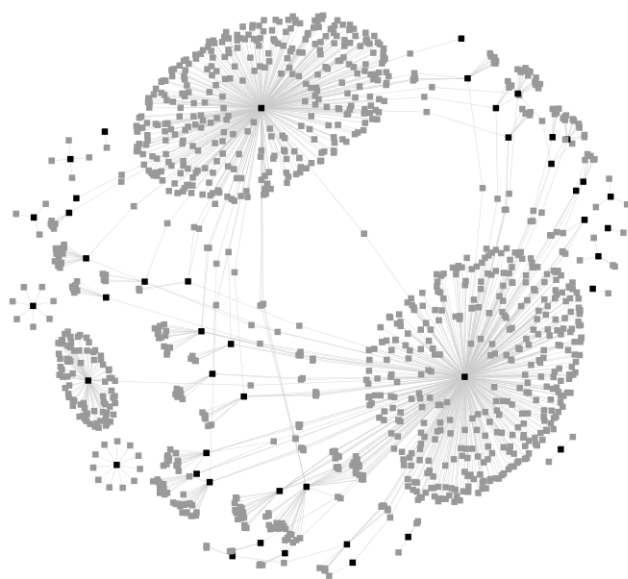
```

1 // Retrieve and collect reactions information
2 foreach($_compounds as $id => $compound){
3     $ec_list = $compound->enzymeIdCollection;
4
5     if( $ec_list ){
6         foreach( $ec_list as $ec ){
7             $ec_id = trim($ec);
8             $ec_loader = new MetaboX\Resource\Loader\
9                 Enzyme($ec_id, $ecLoaderConfig);
10             $enzymes[$ec_id] = $ec_loader->load();
11         }
12     }

```

**Listing 2.** Loading Enzymes metadata from KEGG

The modeling of this class of networks allows to detect which compounds are directly connected, being reagents of the same reactions. It highlights what are the highly connected hubs in a network made up of the collection of metabolites under analysis. This information is useful for planning metabolic



**Figure 1.** Enzyme-metabolite bipartite network: 1196 nodes and 1256 interactions. Darker nodes represent metabolites.

engineering strategies. It is clear that if we wish to modify a node of this type of network, it is crucial to know what are other reactants to be considered, so that the change can effectively impact on the metabolic system of the studied biological organism.

## 2.3 Bipartite Enzyme-Metabolite Network

A network of enzymes and metabolites is a bipartite undirected graph  $Z = (U, V, E)$  with set of nodes  $U$  representing metabolites and  $V$  representing enzymes. A metabolite node is connected to all the enzymes nodes that catalyze a reaction involving that metabolite, and an enzyme node is connected to all the metabolites that take part in the corresponding reaction. That is, if an enzyme  $F$  in  $V$  catalyzes a reaction where a metabolite  $M$  in  $U$  is a substrate, then an interaction between  $F$  and  $M$  exists in the network. We achieve this task using *EnzymeBipartiteGraph* class which parameters are: a metabolite collection, an enzyme collection and a reaction collection. We cycle through the list of metabolites and select the related enzymes. We search current metabolite  $M$  in the substrates of the reaction catalyzed by enzyme  $F$ . If we have a match, we connect nodes  $F$  and  $M$ . An enzymes network, both unipartite and bipartite, provides a kind of visualization that highlights some aspects that are not observable by a reactants network. If we are analyzing different time conditions with different concentration levels of some compounds, for instance, this class of networks would quickly identify which nodes are most affected, restricting the area of interest to the enzyme directly susceptible to a particular condition. Therefore, the construction of this type of graphs can help highlight changes in the enzymatic expression levels or to detect enzymes with structural or functional defects due to particular conditions of stress. An example of such a network is shown in Fig. 1.





**Figure 2.** Network of all metabolites: we were able to find 3049 interactions among 1629 metabolites. Darker nodes represent 48 input metabolites out of 50.

## 2.4 Unipartite Enzymes Network

A unipartite network of enzymes is an undirected graph  $G = (V, E)$  where nodes represent enzymes and two enzymes sharing a common compound in the corresponding reactions are connected to each other. The class used to model such a network is *EnzymeUnipartiteGraph*. This builder class is instantiated with a list of enzymes and a list of reactions. These lists are created collecting all reactions and enzymes that involve input metabolites (Listing 2). For each enzyme in the collection, we load data of the reaction catalyzed and select all substrates. We cycle through the enzyme collection comparing the current enzyme substrates to all others. Given two enzymes  $T$  and  $S$  in  $V$ , we connect them if the intersection between substrates in  $T$  and substrates in  $S$  is not empty. An example of such a network is shown in Fig. 3.

## 2.5 Data Export

In the MetaboX library, there are two classes that can be used to export the constructed network in other formats. As for the other components, a *AbstractGraphWriter* is an abstract class that exposes an abstract *write* method. The class constructor takes one parameter, that is a multidimensional array containing the node list and the weighted edgelist of the network. This can be set using *getGlobalGraph* or *getSubGraph* to export respectively a network or a sub network. The *write* method has two parameters: the name of the file to be written and the data that needs to be exported. If the output needs to be prepared or modified somehow, it is possible to call *prepareOutput* within the *write* method. This

is the case of *CytoscapeGraphWriter* class where interactions are converted to string and then written to file. To work with the D3JS (<http://d3js.org>) visualization library as well as D3py (<https://github.com/mikedewar/d3py>) or NetworkX (<http://networkx.github.io>), the MetaboX library provides several classes to export a network in one of the formats accepted by other analysis tools. For instance, a *D3JSGraphWriter* converts the network to JSON and writes it to file.

## 3. Results and Discussion

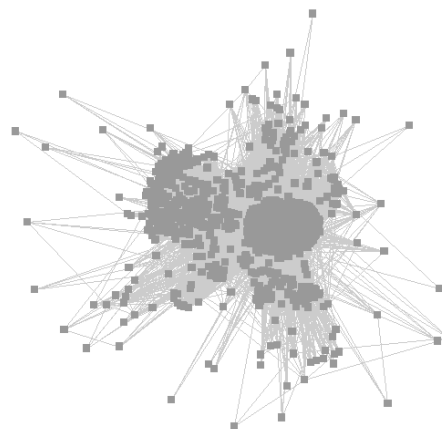
In order to test MetaboX, we built a network starting from a set of 50 *Drosophila melanogaster* metabolites. These compounds are coming from two different genotypes of *D. Melanogaster*, the first one tolerant to thermal shock and the other one susceptible to thermal shock. This second class includes *Drosophila* that need some time to return to a normal state after a thermal shock. For each *D. melanogaster* genotype, we have analyzed three sets of compounds:

- A set of compounds obtained in normal conditions.
- A set of compounds obtained during a thermal shock.
- A set of compounds obtained after a thermal shock.

Fig. 4 shows the network built out of the input metabolites, and we can see that only 37 out of 50 compounds have a direct interaction, featuring 65 total interactions. An interaction in this graph means to be reactants of the same reaction, therefore in this network each node will be directly affected by a

change in one of its neighbors. This information can be useful if we want, for instance, to plan a change in the levels of a compound concentration starting from other compounds already known. ATP (C00002) and NAD<sup>+</sup> (C00003) are highly connected nodes in the network shown in Fig. 4, in fact the first one is considered to be the key molecular unit of intracellular energy transfer and the second one is a coenzyme involved in most of the redox reactions. Fig. 2 shows a network built out of all reactions involving input metabolites. This results in a network that includes input metabolites as well as others related to them. Here we found 3049 interactions between 1629 metabolites; only 48 out of 50 input metabolites are involved in this network. This network provides us a complete view of what the network in Fig. 4 shows only for particular metabolites. From that graph we identify which are the hubs, namely Water, ATP, NAD<sup>+</sup>, NADH, Oxygen and 2-Oxoglutarate. A change in these highly connected nodes may severely affect the *Drosophila* metabolism. Another example we provide is the construction of a unipartite or bipartite enzymes network. In the first case (Fig. 3), a network consists of enzymes and two of them are connected if they share at least one metabolite in the reactions they catalyze, with the constraint that the substrate of one enzyme is the product of another enzyme. We added this constraint to allow the user to easily have a view of the substrate-product flow, looking to enzymes and not to compounds. In this way we were able to build a unipartite enzymes network with 1635 nodes and 141636 interactions. Finally, we build a bipartite enzyme-compound network (Fig. 1). As already mentioned, this network consists of two sets of nodes: compounds and enzymes. Nodes are connected alternatively, that is a compound to an enzyme and vice-versa. Connections between two compounds or two enzymes are not possible. As a resulting network, we were able to find 1196 nodes (45 metabolites and 1151 enzymes) and 1256 interactions. Looking at this network, we easily identify the hubs (ATP, NADP, NADH and L-glutamate) and all the enzymes related to them. To change something in these hubs, the variables (enzymes) to be considered are really a lot, and the design of a subsequent experiment in metabolic engineering would be too complicated. The better solution is, instead, to focus attention on the compounds with few connections, in order to limit the analysis to few enzymes and to decrease the complexity of the further analysis. Anyhow, it is clear how this type of network significantly simplifies the work of those who analyze metabolic pathways to understand metabolic disorders, to connect diseases to enzyme defects, to design successful metabolic engineering strategies.

With the aid of the MetaboX library, we tried to detect differences in metabolism of the two *D. melanogaster* genotypes analyzed. The starting assumption is that if a compound is present in many pathways and reactions, a change in its concentration will dramatically affect the metabolism of the organism. Analysis made in normal condition revealed some constitutive differences between tolerant and susceptible genotypes. Table 1 summarizes the obtained results in



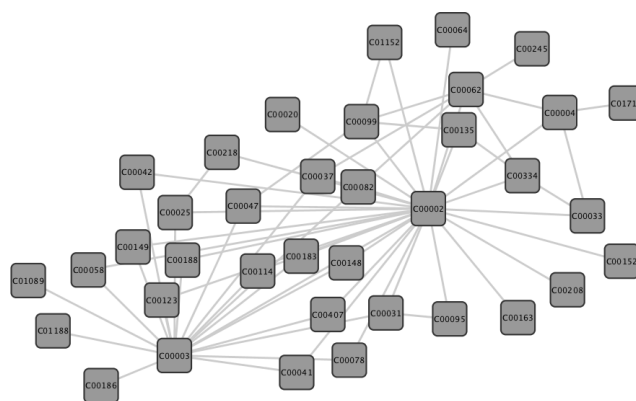
**Figure 3.** Enzymes unipartite network: 1635 nodes and 141636 interactions

terms of percentage of variation of the compounds concentration. In normal condition, tolerant and susceptible differ mainly in the production of 3-hydroxykynurenine (C03227), dimethylamine (C00543) and methylamine (C00218). By observing the reactants network and the bipartite enzyme-metabolite network in Fig. 2 and Fig. 1, we realize that these compounds are not highly connected nodes. In particular, from the bipartite enzyme-metabolite network we can see what are the enzymes that catalyze the reactions of these compounds, obtaining the information about the pathways involved. In this way we learned that differences in the production of 3-hydroxykynurenine highlights differences in the tryptophan metabolism, while a change in the methylamine and dimethylamine production reveals differences in the carbon metabolism. After a thermal shock we observed in tolerant *Drosophila* a significant decrease in the production of methylamine and dimethylamine while in susceptible *Drosophila* a major decrease in the production of 2-hydroxyisobutyrate (C01188) and proline (C00148) was observed. Again we obtained the information about enzymes and pathways from the networks built with the MetaboX library, obtaining that regarding the tolerant genotype, the metabolism of carbon is once again involved, while in the susceptible genotype the involved pathways are the aminoacyl-tRNA biosynthesis, the arginine and proline metabolisms and the ABC transporters pathway. We can conclude that the constitutive differences between tolerant and susceptible can lead to different behaviors in stress adaption. In tolerant *Drosophila* the increased production of methylamine and dimethylamine is balanced by their consumption after the thermal shock. In susceptible *Drosophila* instead, where the starting levels of methylamine and dimethylamine are lower than the other genotype, the organism tries to respond to the shock by increasing the levels of 3-hydroxykynurenine. We don't know if the increased production of methylamine and dimethylamine in the tolerant genotype is due to an increased expression of the enzymes that catalyze the reaction in which these compounds are involved,

or if there is a block in another metabolic pathway that leads to prefer the path that produce methylamine and dimethylamine. Further experiments of metabolic engineering are needed to answer to this question, and the networks we created give a significant help in the design of such experiments. Anyway, the fact that the tolerant and susceptible differ mainly in the production of compounds with few connections in these networks is definitely an advantage, and not an handicap, because it limits the further analysis to few metabolic pathways and to few enzymes. Moreover, in tolerant the only compound that increases after the shock is NADH (C00004), while a decrease is observed for all the compounds. In susceptible instead NADH and isopropanol (C01845) increase. It is interesting to see how NADH concentration increases despite the thermal shock. Analyzing the network that we built, we can see that NADH is always a highly connected node, so it should be a key compound in the *Drosophila* metabolism. It must be remembered that *D. melanogaster* susceptible genotype that we observed corresponds in period of time to return in a normal state after a thermal shock, therefore the shock does not result in a dramatic effect. Probably if we observed a change in the concentration of NADH between the two genotypes, it would result in more severe consequences than those observed. In conclusion, if we start without any initial information about compounds concentration and we look at a general network, for sure the highly connected nodes are fundamental in the metabolism, and changes in these nodes would have probably led to the death of the organism. On the contrary, if we start from experimental data, it might be useful to correlate increases or decreases of the concentration of some compounds to a particular disease or to a particular disorder. Therefore highlighting compounds within a network should be useful for designing any strategy aimed at clarifying the mode of occurrence of the disease, extracting from that network information like number of edges, enzymes, reactions, etc. The MetaboX library is a suitable tool created to solve both issues: a first preliminary view and a second in-depth analysis.

## 4. Conclusions

At the moment of this writing, the MetaboX library represents the first example in this class of software. Online bio databases such as KEGG do not provide sufficient APIs for extensive and more complex data manipulation. With the Bioinformatics toolbox in Matlab, it was possible to use KEGG SOAP APIs with the ability to reproduce pathways or convert NCBI ids to KEGG ids. This is no longer possible since this kind of APIs have been suppressed. Furthermore, RESTful KEGG interface only provides simple plain text metadata for biological resources. As it is possible to easily build a network of a few nodes manually using tools like Cytoscape, we needed a tool that would create a network programmatically given larger lists of input nodes. MetaboX could use alternative resource providers in order to retrieve nodes and interactions metadata to build different kind of networks. Fi-



## References

- [Cline *et al.* 2007] Cline, M. S. et al. (2007) Integration of biological networks and gene expression data using Cytoscape. *Nat Protoc.*, 2366-82.
- [Cloots *et al.* 2011] Cloots, L. et al. (2011) Network-based functional modeling of genomics, transcriptomics and metabolism in bacteria. *Curr Opin Microbiol.*, 599-607.
- [Fielding *et al.* 2002] Roy, T. F. et al. (2002), Principled Design of the Modern Web Architecture. *ACM Transactions on Internet Technology*, 115-150.
- [Franceschini *et al.* 2013] Franceschini A, Szklarczyk D, et al. (2013) STRING v9.1: protein-protein interaction networks, with increased coverage and integration. *Nucleic Acids Res* 41(Database issue): D808-815.
- [Hagberg *et al.* 2008] Hagberg, Aric A. et al. (2008) Exploring Network Structure, Dynamics, and Function using NetworkX. *Proc. SciPy 2008*, 11-16.
- [Hastings *et al.* 2013] Hastings, J. et al. (2013) The ChEBI reference database and ontology for biologically relevant chemistry: enhancements for 2013. *Oxford Journals Nucl. Acids Res.*, D456-63.
- [HU *et al.* 2011] HU, K. et al. (2011) Metabolic network modeling and simulation for drug targeting and discovery. *Biotechnol J.*, 3, 30-42.
- [Kanehisa *et al.* 2011] Kanehisa, M. et al. (2011) KEGG for integration and interpretation of large-scale molecular data sets. *Nucl. Acid Res.*, (Database issue) D109-14.
- [Karp *et al.* 2005] Karp, P. D. et al. (2005) Expansion of the BioCyc collection of pathway/genome databases to 160 genomes. *Nucl. Acid Res.*, 6083-6089.
- [Keseler *et al.* 2013] Keseler, I. M. et al. (2013) EcoCyc: fusing model organism databases with systems biology. *Oxford Journals Nucl. Acids Res*, (Database issue) D605-12.
- [Krieger *et al.* 2004] Krieger, C. J. et al. (2004) MetaCyc: a multiorganism database of metabolic pathways and enzymes, *Oxford Journals Nucl. Acids Res*, D511-D516.
- [Mak *et al.* 2013] Mak, T. D. et al. (2013) MetaboLyzer: A Novel Statistical Workflow for Analyzing Postprocessed LC-MS Metabolomics Data. *Anal. Chem.*, Article ASAP, 506-13.
- [Matthews *et al.* 2009] Matthews, L. et al. (2009) Reactome knowledgebase of human biological pathways and processes. *Nucl. Acids Res.*, (Database issue) D619-22.
- [Menikarachchi *et al.* 2013] Menikarachchi, L. C. et al. (2013) In Silico Enzymatic Synthesis of a 400.000 Compound Biochemical Database for Nontargeted Metabolomics. *J. Chem. Inf. Model.*, 2483-2492.
- [Pence *et al.* 2010] Pence, H. E. et al. (2010) ChemSpider: An Online Chemical Information Resource. *J. Chem. Educ.*, 1123-1124.
- [Raosaheb *et al.* 2005] Raosaheb, K. et al. (2005) Uncovering transcriptional regulation of metabolism by using metabolic network topology, *PNAS*, 2685-2689.
- [Sharma *et al.* 2014] Sharma A et al (2014) Rigidity and flexibility in protein-protein interaction networks: a case study on neuromuscular disorders, *arXiv:1402.2304v2*
- [Smith *et al.* 2005] Smith, C. A. et al. (2005) METLIN: A Metabolite Mass Spectral Database Therapeutic Drug Monitoring. *Proc. of the 9th ICTDM*, 747-751.
- [Sud *et al.* 2007] Sud, M. et al. (2007) LMSD: LIPID MAPS structure database. *Oxford Journals Nucl. Acids Res*, 527-32.
- [Wang *et al.* 2012] Wang, Y. et al. (2012) PubChem's BioAssay Database. *Nucl. Acids Res.*, D400-D412.
- [Wishart *et al.* 2007] Wishart, D. S. et al. (2007) HMDB: the Human Metabolome Database. *Nucleic Acids Res*, (Database issue) D521-6.
- [Xia *et al.* 2012] Xia, J. et al. (2012) MetaboAnalyst 2.0 - a comprehensive server for metabolomic data analysis. *Nucl. Acids Res.*, 1-7.
- [Xia *et al.* 2013] Xia, J. et al. (2013) INMEX—a web-based tool for integrative meta-analysis of expression data. *Nucl. Acids Res.*, (Web Server issue) W63-70.



**Table 1.** Compounds name and KEGG IDs are shown in the first two columns; the percentage variation of each compound concentration between the tolerant genotype and the susceptible genotype of *Drosophila melanogaster* is shown in the third column; the percentage variation of each compound concentration between the state after and before the thermal shock in tolerant *Drosophila* is shown in the fourth column; the percentage variation of each compound concentration between the state after and before the thermal shock in susceptible *Drosophila* is shown in the fifth column.

Compound	KEGG ID	% Tol./Susc. Time 0	% Time 1/Time 0 Tolerant	% Time 1/Time 0 Susceptible
Leu	C00123	13.85	-16.97	-20.37
Val	C00183	0.33	-19.54	-26.84
Ile	C00407	3.45	-16.45	-22.80
Propionate	C00163	25.40	-42.19	-18.41
Hydroxyisovalerate	C04272	28.37	-19.95	-22.16
Lactate	C00186	27.44	-28.09	-17.30
Thr	C00188	36.22	-10.78	-29.11
<b>2-Hydroxyisobutyrate</b>	C01188	-5.93	-18.25	<b>-50.47</b>
Ala	C00041	3.01	-16.25	-23.06
Acetate	C00033	5.23	-23.53	-21.24
<b>Pro</b>	C00148	-8.10	-45.08	<b>-51.52</b>
Glu	C00025	1.72	-33.93	-46.53
Succinate	C00042	22.37	-17.94	-21.71
B-Ala	C00099	19.85	-25.85	-31.72
Malate	C00149	-10.17	-23.48	-42.38
<b>Dimethylamine</b>	C00543	<b>69.47</b>	<b>-59.99</b>	<b>-29.76</b>
Sarcosine	C00213	33.62	-23.58	-41.58
Asn	C00152	-9.05	-1.71	-26.61
Choline	C00114	36.74	-16.46	-14.16
Phosphorycholine	C00588	49.89	-27.12	-28.92
Gly	C00037	9.66	-22.59	-21.96
Fructose	C00095	38.08	-27.87	-23.52
Trehalose	C01083	13.80	-23.18	-28.11
Sucrose	C00089	-27.79	-40.21	-30.62
Glucose	C00031	3.06	-17.99	-10.73
Fumarate	C00122	8.87	-32.70	-39.61
<b>3-hydroxykynurenine</b>	C03227	<b>-91.45</b>	-19.97	-37.93
Kynuresine	C00328	31.29	-31.00	-21.05
Tyr	C00082	-49.69	-24.90	-22.80
Kynurenate	C01717	17.40	-11.68	-12.85
His	C00135	39.17	-28.66	-26.93
NAD+	C00003	4.52	-24.95	-26.04
Formate	C00058	14.35	-12.71	-19.04
AMP	C00020	13.88	-30.23	-22.40
ATP/ADP	C00002	-1.84	-41.81	-46.30
3-hydroxybutyrate	C01089	26.59	-29.62	-34.23
4-aminobutyrate	C00334	14.79	-26.86	-38.25
Acetamide	C02505	-10.15	-35.88	-46.32
Arg	C00062	7.05	-27.37	-31.22
Gln	C00064	4.04	-9.75	-6.37
Isopropanol	C01845	46.79	-12.10	25.70
Lys	C00047	23.45	-22.78	-34.82
Maltose	C00208	-12.36	-17.55	-37.93
<b>NADH</b>	C00004	39.44	21.11	36.88
<b>Methylamine</b>	C00218	<b>69.18</b>	<b>-68.00</b>	-18.63
t-methylHis	C01152	19.15	-14.13	-11.93
n-methylHis	C03298	33.27	-33.11	-12.59
Trp	C00078	16.50	-21.19	-29.21
Taurine	C00245	-7.51	-22.71	-32.72
Ornithine	C00077	13.04	-21.18	-32.06